SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

Sustav udaljenog upravljanja vozilom s povratnom video vezom

Završni rad

Leonardo Markotić

Osijek, 2018.

Sadržaj

1. UVOD

Pojam udaljenog upravljanja spominje se još davne 1898. godine kada je Nikola Tesla prvi put pokazao ideju broda upravljanog radio vezom. Brod se sastojao od upravljačke ploče, odnosno odašiljača i upravljanog objekta, odnosno prijamnika. Pomoću radio valova signali su se slali s odašiljača na prijamnik i tako se upravljalo *coherer-om*, uređajem koji je prenosio radio valove u mehaničke pokrete broda.

Od tada pa sve do danas princip je ostao isti. Koriste se radio signali kao kod Teslinog broda, a za napajanje se koriste baterije. Današnji svijet je nezamisliv bez takvih uređaja. Nalaze se u svakodnevnoj ljudskoj okolini, a primjenjuju se u kućanstvima pa sve do vojnih svrha. Čovjek svakodnevno sve više teži k tome da sve procese oko sebe upravlja na daljinu kako bi sebi olakšao posao, ali i udaljio sebe od procesa kako bi se zaštitio.

Svakodnevnim razvojem tehnologije te naprednijim sustavima možemo reći da će daljinsko upravljani automatizirani procesi zamjeniti čovjeka. Zbog bržeg vremena odziva i vremena reakcije od čovjeka, upravljani procesi su svakoga dana sve točniji i pouzdaniji. Povećanjem sigurnosti i učinkovitosti, čovjek se sve više oslanja na takve sustave. Takvi sustavi mogu spriječiti neku vrstu štete, odnosno mogu prepoznati što je dobro za sustav, a što ne pa možemo reći da ima neki oblik umjetne inteligencije – pomoću senzora i ostalih uređaja sposobni su sami prepoznati prepreke i pronaći rješenje kako bi se sustav čim prije vratio u normalan rad.

Potrebno još puno vremena da se takvi sustavi dovedu do savršenstva i da se njihov rad može izjednačiti s radom čovjeka, ali u današnjem vremenu i s današnjom tehnologijom to je dovedeno do maksimuma.

Tema ovog završnog rada je sustav udaljenog upravljanja vozila s povratnom video vezom. Rad obuhvaća 5 poglavlja u kojima se govori o principu rada te izradi samog modela i programa. Zadnje poglavlje rada sadržava testiranje točnosti sustava te vremena odziva.

1.1. Zadatak i struktura završnog rada

U ovom završnom radu radit će se sustav udaljenog upravljanja vozilom s povratnom video vezom pomoću Raspberry Pi mikroračunala. Sustav će pomoću Raspberry Pi mikroračunala biti povezano s računalom i time ostvariti komunikaciju i način upravljanja, a isto tako sustav će imati mogućnost samoupravljanja, odnosno pomoću dodatne kamere ostvarit će *line tracking*, način upravljanja u kojem će sustav prepoznati zadanu crtu te će se vozilo kretati po toj crti. Za video povratnu vezu koristi se analogna mini fpv (engl. *First person view*) kamera i njen prijamnik koji rade na frekvenciji od 5.8gHz tako da će kašnjenja biti minimalna.

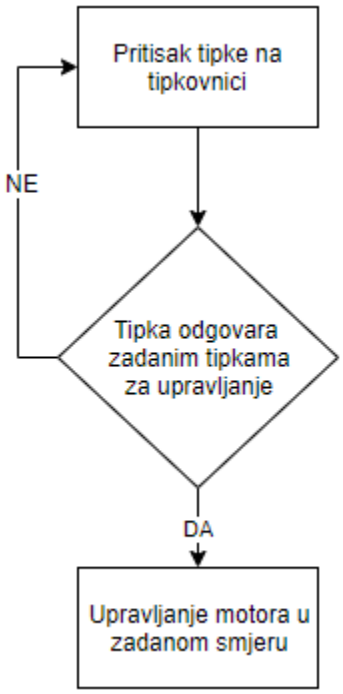
2. DALJINSKO UPRAVLJANJE VOZILOM I POVRATNA VIDEO VEZA

Kao što je već spomenuto, u ovom radu za udaljeno upravljanje koristit će se Raspberry Pi mikroračunalo. Na Raspberry Pi-u pomoću Linux operacijskog sustava se spaja na lokalnu mrežu. Tom komunikacijom omogućavamo upravljanje preko računala. Za daljinsko upravljanje koristit će se tipkovnica osobnog računala. Jednostavnim prepoznavanjem tipke motori će se kretati u zadanome smjeru.

Line tracking je ipak nešto kompliciraniji jer zahtjeva detaljnu obradu slike. Koristit će se dodatna kamera koja će slati slike (engl. *Frame*) na Raspberry Pi (odprilike 5 slika u sekundi). Zatim će Raspberry Pi tražiti određenu nijasnu boje te će ovisno o položaju crte usmjeravati svoje motore.

Video povratna veza sastoji se od analogne kamere koja će odašiljati sliku te analognog prijamnika koji će primati sliku. Poželjno je da kašnjenja budu minimalna, tj. skoro zanemariva pa se koristi visoka frekvencija u iznosu od 5.8. gHz.

2.1. Daljinsko upravljanje vozilom



Slika 2.1. Prijedlog algoritma za udaljeno upravljanje

Na slici 2.1. prikazan je prijedlog algoritma za udaljeno upravljanje. Ideja je veoma jednostavna. Pritiskom određene tipke na tipkovnici upravljat će se određeni motor. Ukoliko pritisnuta tipka ne odgovara ni jednoj postavljenoj tipci za upravljanje algoritam će se ponavljati sve dok se ne pritisne zadana tipka. Zbog toga što se projekt sastoji od dva programa, potrebno je izraditi neki oblik *software-ske* sklopke koja će omogućiti promjenu programa.

2.2. Senzori i aktuatori

Kao što je već spomenuto, za detekciju prepreka koristi se senzor. Na slici 2.2. prikazan je pricip rada ultrazvučnog senzora. Senzor se sastoji od prijamnika i odašiljača. Odašiljač šalje

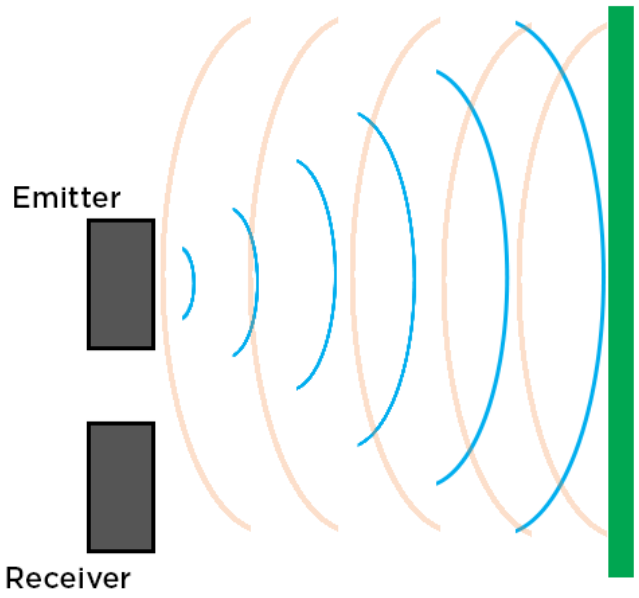
valove brzinom svjetlosti (v = 340 $^{m}/_{s}$) te kada se se ti valovi odbiju od predmeta vraćaju se do prijamnika te se mjeri vrijeme trajanja valova. Prema formuli 2-1 računa se udaljenost predmeta od senzora.

$$s = \frac{t*v}{2} \qquad (2\text{-}1)$$

gdje je: s – udaljenost predmeta od senzora

t – vrijeme potrebno valu da od odašiljača dođe do prijamnika

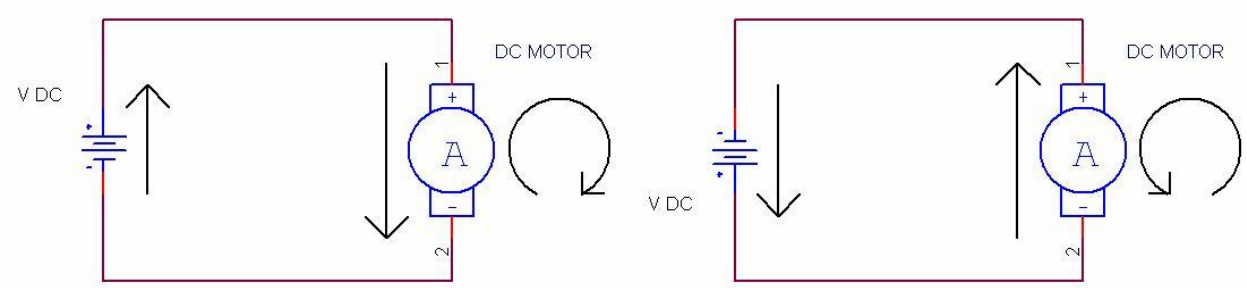
v – brzina svjetlosti (340 $^{m}/_{s}$)



Slika 2.2. Princip rada senzora

Aktuatori su uređaji koji pomoću vanjske pobude svoje pokretne dijelove dovode u željeni položaj, tj. oni neku ulaznu veličinu pojačavaju i pretvaraju u mehanički rad. U ovom slučaju to su istosmjernu motori. Pomoću impulsa s Raspberry Pi-a oni će se okretati u zadanome

smjeru. Upravljački pinovi spajat će se na GPIO pinove Raspberry Pi-a. Za svaki motor koristit će se dva upravljačka pina jer postoje dvije kombinacije, kada je jedan pin u logičkoj „1“, a drugi u logičkoj „0“ tada će se motor okretati u jednom smjeru, a ukoliko se logička „1“ i logička „0“ zamjene, motor će se okretati u drugom smjeru.



Slika 2.3. Smjer vrtnje DC motora

Na slici 2.3. prikazan je način rada DC motora. Vidljivo je da, kao što je već spomenuto, smjer vrtnje ovisi o polaritetu izvora. Ukoliko se promjeni polaritet, promjenit će se i smjer vrtnje motora.

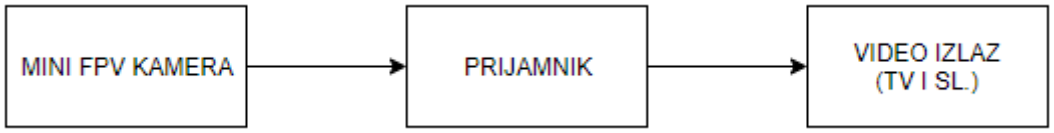
2.3. Algoritam upravljanja

Slika 2.4. Prijedlog algoritma za *line tracking*

Na slici 2.4. prikazan je prijedlog algoritma za *line tracking*. Vidimo da sve ovisi o komunikaciji između kamere i Raspberry Pi-a. Raspberry Pi dohvaća sliku s kamere te ju zatim obrađuje, odnosno „traži“ sve što se nalazi u zadanome raspnu boja te ovisno o tome usmjerava motore RC modela. Ukoliko Raspberry Pi ne može naći ništa što se nalazi u rasponu boja, algoritam se vraća na početak, a motori miruju.

2.4. Video veza i integracija u sustav

Video veza je osnovni dio sustava. Omogućava da prilikom udaljenog upravljanja dobivamo živu sliku. Na slici 2.5. prikazan je blok dijagram rada video veze.



Slika 2.5. Blok dijagram video veze

Kao što je već spomenuto, video veza sastoji se od dva dijela, odašiljača i prijamnika, pri tome da prijamnik mora biti spojen na neki izlaz kao što je TV. Video veza je veoma važna jer moguće je da prilikom udaljenog upravljanja neće biti vizualnog dodira s RC modelom, pa se pomoću povratne video veze može udaljeno upravljati.

3. REALIZACIJA SUSTAVA

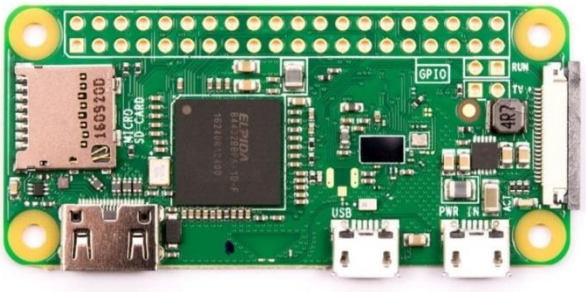
Kako bi se ovaj zadatak ostvario, potrebno je mnogo elektroničke opreme. U ovom poglavlju nabrojat će se osnovna oprema, te će se za istu dati kratki opis, opisati će se postupak izrade samog modela te izrada programa.

3.1. Korišteni alati i programsko sučelje

Popis opreme:

- Raspberry Pi Zero W
- Ultrazvučni senzor HC-SR04
- 5.8g fpv (engl. *First person view*) mini kamera
- 5.8g prijamnik
- Web kamera
- Platforma za izradu
- Akumulatorska baterija Emos 12V, 4.5Ah, Model: OT 4.5-12
- Regulator HCJ-IPM-V2
- Driver L298N

3.1.1. Raspberry Pi Zero W



Slika 3.1. Raspberry Pi Zero W

Raspberry Pi Zero W nastao je kao dodatak Raspberry Pi obitelji. Na prethodni Raspberry Pi Zero model dodan je integrirani *wifi* modul te je time postao najpopularniji Raspberry Pi model zbog svojih specifikacija i cijene. Sadrži dva „mikro USB“ ulaza, jedan „mini HDMI“ ulaz te sadrži mjesto za „mikro SD“ karticu. U sebi ima intergrirani *wifi* i *Bluetooth* modul. Sa svojim 1 GHz procesorom i 512 MB RAM-om, napajanjem od 5V te pogonjeno Linux operacijskim sustavom, Raspberry Pi Zero W odlično je računalo za ovaj zadatak.

3.1.2. Ultrazvučni senzor HC-SR04

Slika 3.2. Ultrazvučni senzor HC-SR04

Ultrazvučni senzor HC-SR04 sastoji se od ultrazvučnog odašiljača, ultrazvučnog prijamnika i upravljačke jedinice. Odašiljač šalje ultrazvučne valove frekvencijom 40 kHz i ukoliko se nalazi neka prepreka valovi se odbijaju te se vraćaju u prijamnik. Prednosti ovog senzora su te što ga ne ometaju sunčeva svjetlost, crni materijali ili neke druge tvari. Ima domet mjerenja 2-500 cm s preciznošću 0.03 cm.

3.1.3. 5.8g fpv mini kamera

Slika 3.3. 5.8g fpv mini kamera

Analogna kamera minijaturnih dimenzija, a visokih perfomansi savršena je za ovakav projekt. Služi za prijenos žive slike (engl. *Live stream*). Zbog visoke frekvencije odašiljanja, kašnjenje slike biti će minimalno.

3.1.4. 5.8g prijamnik

Slika 3.4. 5.8g prijamnik

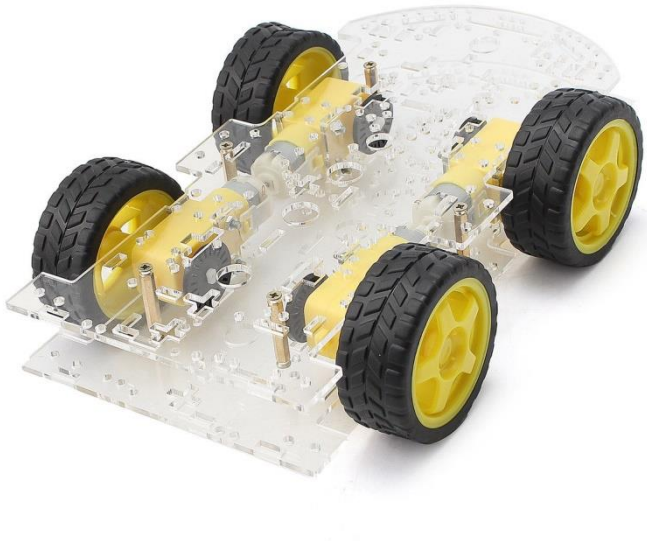
Analogni prijamnik koji će služiti za primanje signala koji će mini kamera odašiljati. Koristi analogni izlaz tako da će biti spojen na vanjski uređaj (TV ili slično). Ima domet od 5 km u otvorenom području tako da neće biti smetnji.

3.1.5. Web kamera

Slika 3.5. Web kamera

Ova kamera služi za *line tracking*. Kamera će slati 5 slika u sekundi na Raspberry Pi te će on tada obraditi sliku i zaključiti gdje se nalazi zadana crta, a gdje se nalazi RC model te ovisno o tome upravljati RC modelom. Kameru na Raspberry Pi spajamo pomoću USB kabla.

3.1.6. RC model na daljinsko upravljanje



Slika 3.6. Platforma za izradu

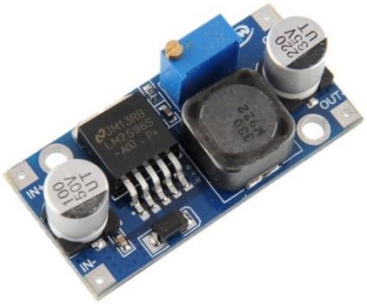
Za izradu projekta koristi se gotova platforma za izradu. Na njoj se nalaze motori s izvodima i kotači. Motori su spojeni u paralelu, odnosno motori koji se nalaze na lijevoj strani platforme su spojeni u paralelu te motori koji su s desne strane platforme također su spojeni u paralelu. Na motorima se već nalaze izvodi koji se kasnije spajaju na driver.

3.1.7. Akumulatorska baterija Emos

Slika 3.7. Akumulatorska baterija

Za napajanje projekta koristi se akumulatorska baterija Emos nominalne snage 4500mAh i 12V. Zbog svojih malih dimenzija, a visokog kapaciteta odlična je za ovaj projekt. Akumulatorska baterija napajati će čitavi sustav, a pomoću regulatora napona ostvarit će se prijelaz s 12V na 5V.

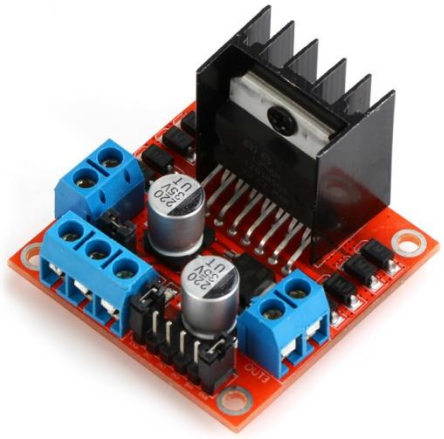
3.1.8. Regulator HCJ-IPM-V2



Slika 3.8. Regulator HCJ-IPM-V2

Zbog toga što akumulatorska baterija daje izlaz od 12V, a Raspberry Pi za napajanje koristi 5V, mora se koristiti regulator kako bi smanjio napon i kako ne bi došlo do oštećenja sustava.

3.1.9. Driver L298N

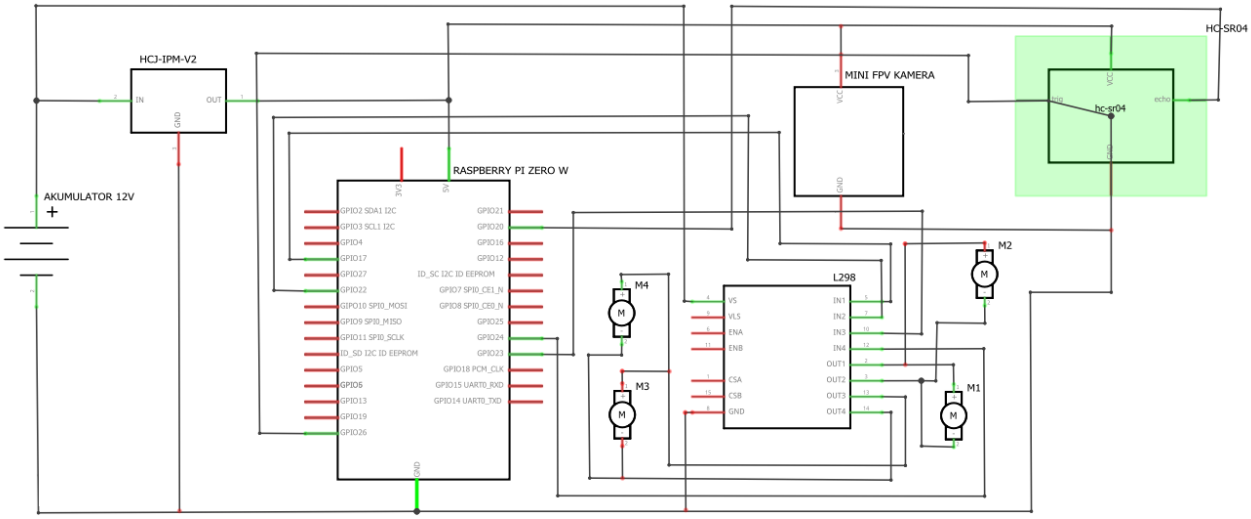


Slika 3.9. Driver L298N

Kao što je već spomenuto, zbog toga što je ulazna struja Raspberry Pi-a odprilike 40mA po pinu, a struja motora iznosi odprilike 200mA na više, moramo koristiti driver-e za motor kako bi tu struju smanjili I zaštitili pinove, a da pritom snaga motora ostane ista. Driver L298N omogućava upravljanje dva motora istovremeno te ima ulazni napon 12-35V.

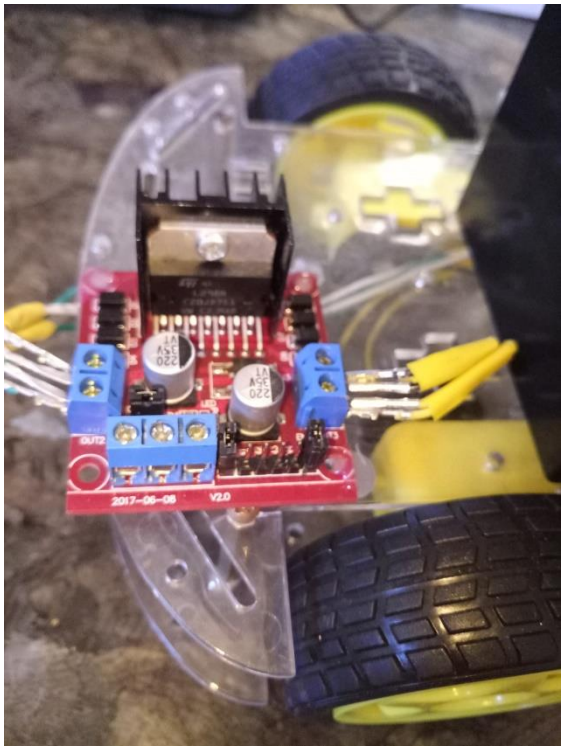
3.2. Izgradnja sustava

Na slici 3.10. prikaza je shema spajanja sustava. Kao što je već spomenuto, za napajnje se koristi akumulator napona 12V. On se spaja u paralelu s regulatorom koji taj napon spušta na 5V te se zatim regulator spaja na Raspberry Pi. Zbog toga što Driver L298N radi pri naponu od 12-35V njega možemo spojiti izravno na akumulator.



Slika 3.10. Shema sustava

Za izradu projekta koristi se gotova platforma s motorima i kotačima. Zbog potrebe za statičnim stanjem komponenata, sve komponente ljepe se silikonom zbog toga što ne oštećuje opremu, a veome je efikasan. Zbog toga što su motorima već dodane žice na izvode, njih je potrebno spojiti na Driver L298N kako bi se osigurala zaštita Raspberry Pi-a.



Slika 3.11. Platforma nakon dodanog driver-a

Nakon što je dodan Driver L298N, potrebno je dodati izvor napajanja. Kao što je već spomenuto, za izvor napajanja koristi se akumulator Emos 12 V, 4.5Ah. Zbog toga što je ulazni napon Raspberry Pi-a 5V, a napon akumulatora iznosi 12V, pomoću regulatora „spuštamo“ napon na 5V.

Slika 3.12. Platforma nakon dodanog napajanja

Web kamera dodaje se na prednji dio platforme kako ne bi imala nikakvih prepreka prilikom praćenja crte. Poželjno je da bude što više nagnuta prema dole kako bi smanjili utjecaj okoline, tj. poželjno je da vidi samo crtu.

Slika 3.13. Platforma s kamerom za *line tracking*

Senzor se dodaje što je niže moguće kako bi mogao detektirati prepreke koje kamere ne vide. U slučaju da senzor detektira prepreku, blokira se rad motora i model miruje. Zbog nepraktičnosti same platforme, nije moguće centrirati senzor pa je smješten na lijevoj strani, ali to ne bi trebalo smetati u radu modela.

Slika 3.14. Platforma nakon dodanog senzora

Mini FPV kamera dodaje se na najviše mjesto kako bi imala najbolju preglednost. Ona ne koristi GPIO pinove već samo napajanje s Raspberry Pi-a.

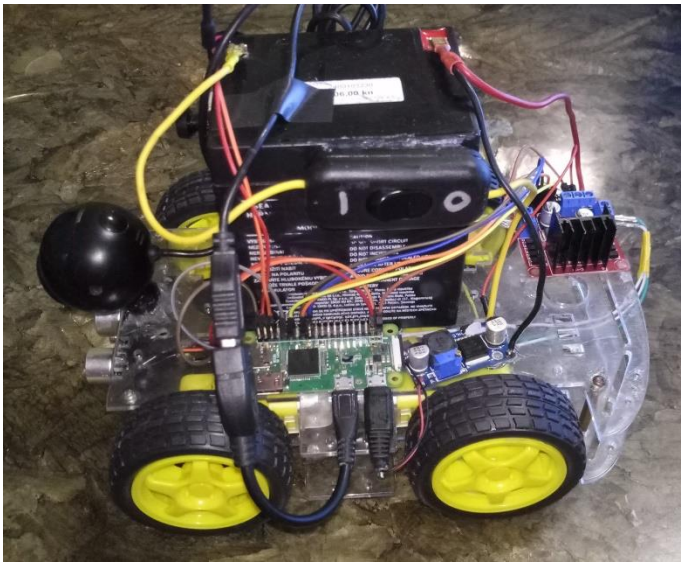
Slika 3.15. Platforma nakon dodane mini FPV kamere

Na kraju se dodaje regulator i Raspberry Pi. Zbog toga što Driver L298N koristi napajanje 12-35 V, on se spaja direktno na akumulator. Na regulator se lemi žica koja „dolazi“ od akumulatora, a s druge strane lemi se micro USB kabel prikazan na slici 3.15. S njega je skinuta izolacija te su uklonjene „data“ žice.

Slika 3.16. Micro USB kabel

Kao zaštitu da sustav ne bi bio pod stalnim napajanjem dodaje se sklopka prikazana na slici 3.16. Na slici 3.17. prikazano je konačno stanje sustava sa svim komponentama i spojevima.

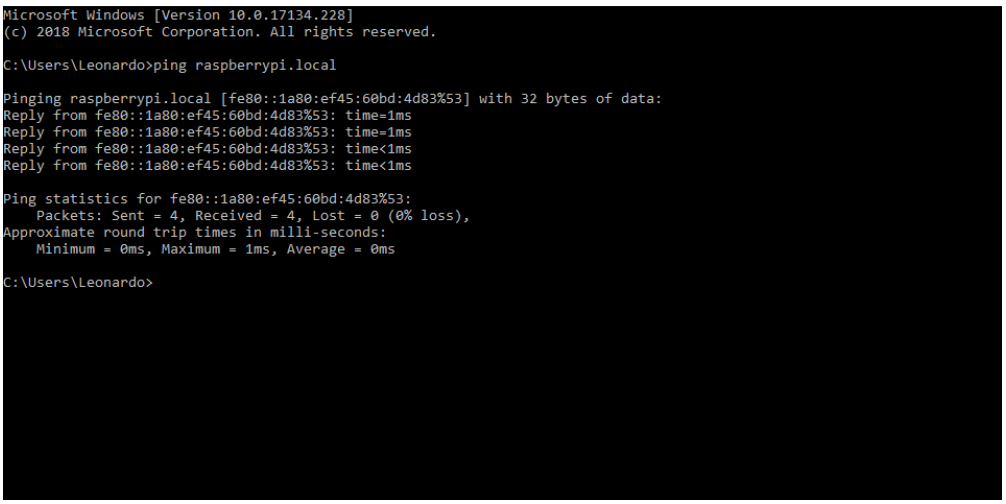
Slika 3.17. Dodavanje sklopke



Slika 3.18. Konačno stanje sustava

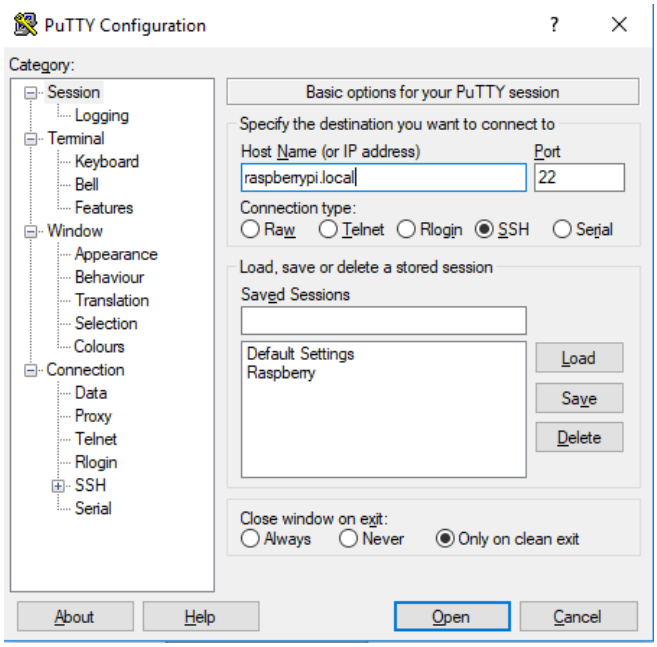
3.3 Izgradnja upravljačkog programa

Prilikom spajanja Raspberry Pi-a na računalo potrebno je napraviti par koraka. Zbog toga što Raspberry Pi Zero W na sebi nema Ethernet priključak kao dodatak, potrebno je preko USB porta ostvarit Ethernet komunikaciju. Kao protok za povezivanje koristio se SSH (Secure Shell) protokol koji korisnicima omogućuje sigurno komunikaciju između dva računala, u ovom slučaju računala i Raspberry Pi-a. SSH protokol radi na principu korištenja kombinacije simetrične i asimetrične kriptografije, odnosno metode enkripcije koja pruža veoma siguran prijenos podataka računalnom mrežom. Nakon toga potrebno je provjeriti da li je konekcija uspostavljena i za to je korišten CMD (Command Prompt). Kao što je vidljivo na slici 3.19. windows šalje 4 paketa podatka i provjerava vezu i vrijeme odziva s Raspberry Pi-om.



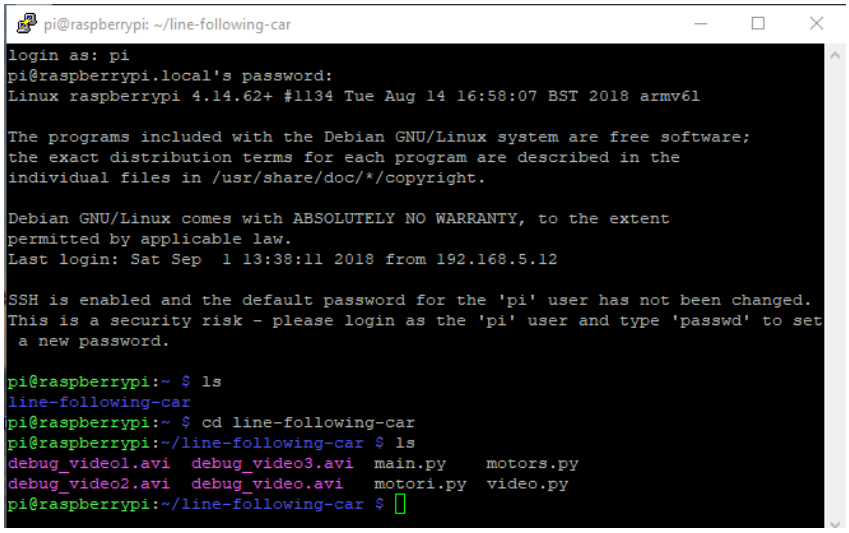
Slika 3.19. Korištenje CMD-a

Nakon uspostavljenje veze, koristi se Putty, konzola koja služi za rad datotekama. Putty podržava mnoge protokole, među kojima je i SSH. Prilikom prijave potrebno je unjeti ime uređaja na koji se spajamo ili njegovu IP adresu.



Slika 3.20. Prijava u Putty

Nakon toga otvara se prozor u kojem upravljamo radom datoteka. Za izradu projekta koristi se Python programski jezik. Pomoću naredbi cd (chdir – change directory) i ls (list) odabiremo datoteke koje želimo koristit. Cd služi kako bi otvorili datoteku, a pomoću naredbe ls izlistavamo sve datoteke koje se nalaze na njoj. Za pokretanje programa koristi se naredba python (npr. python motori.py), a za izmjenu programa koristi se naredba nano (npr. nano motori.py).



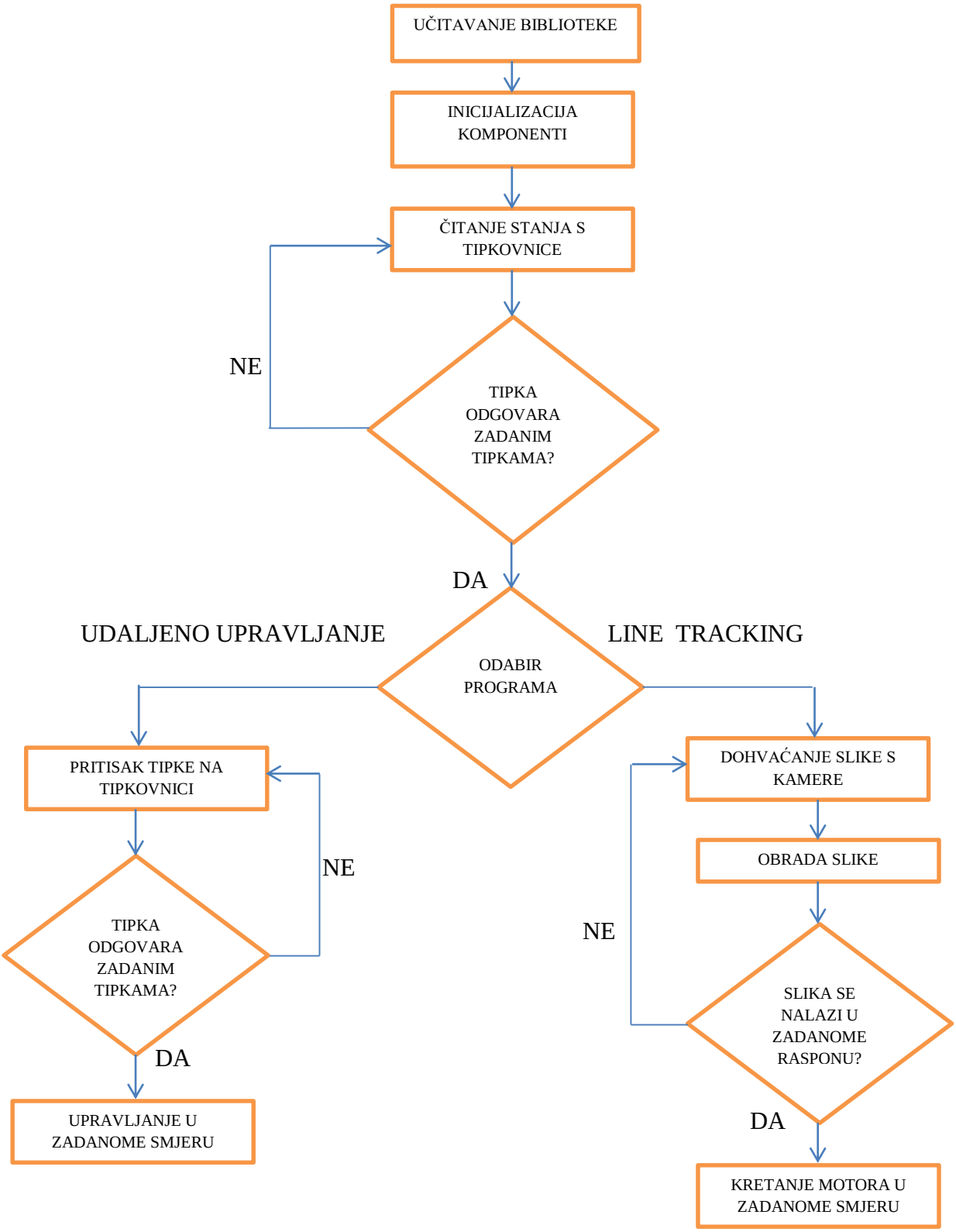
Slika 3.21. Upravljanje datotekama u Putty sučelju

Prilikom izrade glavnog programa bilo je potrebno odrediti pinove na koji su motori spojeni, te smjer u kojem će se vrtjeti. Na slici 3.22. prikazan je testni program gdje je nizom izmjena i kombinacija pinova utvrđeno pri kojoj kombinaciji pinova će motor ići u kojem smjeru. Vidimo da prilikom rada motora jedan logički pin mora biti u logičkoj „1“, a drugi u logičkoj „0“. Ukoliko ih zamjenimo motor će se vrtjeti u suprotnom smjeru.

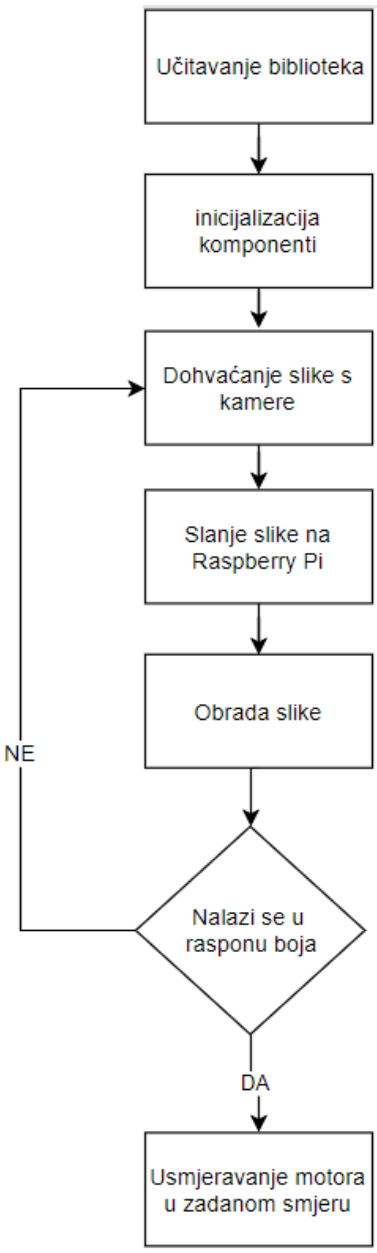
```
def motor_forward():
        gpio.output(LIJEVO_NAPRIJED, True)
        gpio.output(LIJEVO_IZA, False)
        gpio.output(DESNO_NAPRIJED, True)
        gpio.output(DESNO_IZA, False)

def motor_reverse():
        gpio.output(LIJEVO_NAPRIJED, False)
        gpio.output(LIJEVO_IZA, True)
        gpio.output(DESNO_NAPRIJED, False)
        gpio.output(DESNO_IZA, True)

def motor_right():
        gpio.output(LIJEVO_NAPRIJED, True)
        gpio.output(LIJEVO_IZA, False)
        gpio.output(DESNO_NAPRIJED, False)
        gpio.output(DESNO_IZA, True)

def motor_left():
        gpio.output(LIJEVO_NAPRIJED, False)
        gpio.output(LIJEVO_IZA, True)
        gpio.output(DESNO_NAPRIJED, True)
        gpio.output(DESNO_IZA, False)
```

Slika 3.22. Testni program

Glavni dio projekta je „Line tracking“. Web kamera slati će slike na Raspberry Pi (odprilike 5 slika u sekundi) te će ih on obraditi. Na slici 3.23. prikazan je cjelokupni program. Vidljivo je da se program sastoji od dva dijela. Odabir programa omogućen je pristiskom tipke na tipkonici. U ovom slučaju to je tipka „M“. Ukoliko se odabere način udaljenog upravljanja, sustav se bazira samo na čitanje stanja s tipkovnice i komunikacije između modela i računala, a ukoliko se odabere način praćenja crte, sustav se bazira samo na stanje kamere sve dok se radnja ne prekine, odnosno prebaci se na način udaljenog upravljanja. Odabir načina rada moguće je u bilo kojem trenutku.



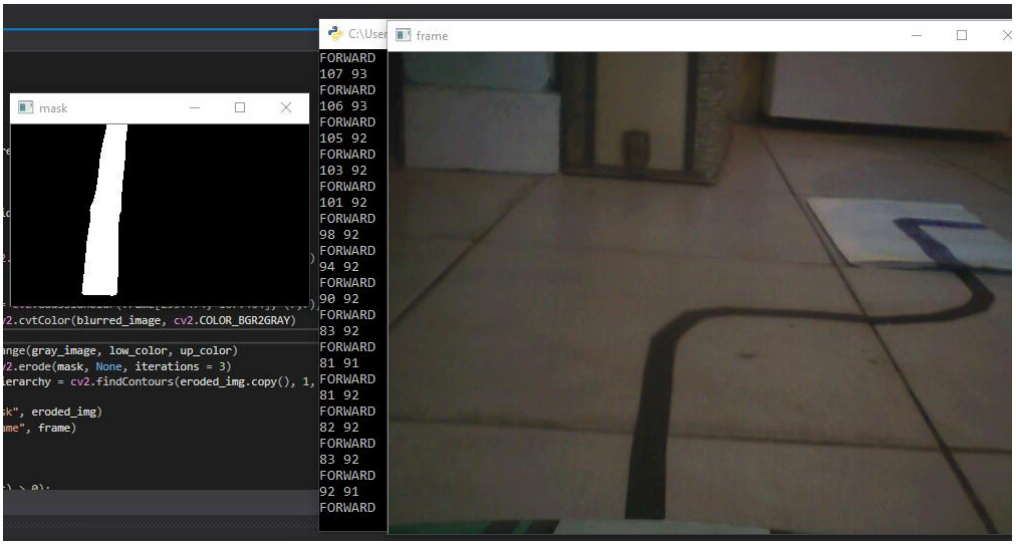
Slika 3.23. Blok dijagram sustava



Slika 3.24. Algoritam za obradu slike

Kako bi omogućili „*line tracking*“ način rada, potrebno je odabrati raspon boja koje će Raspberry Pi obraditi. Kako se raspon boja kreće od 0 do 255, gdje je 0 crna boja, a 255 bijela, za projekt je odabran raspon od 0 do 90, gdje je 90 siva boja. Prilikom obrade slike, Raspberry Pi u obzir uzima sve što se nalazi u tom rasponu. Ukoliko ne može naći ništa u zadanome rasponu,

ponovno dohvaća sliku s kamere kao što je prikazano na slici 3.24. gdje se vidi algoritam rada praćenja crte. Kako bi se smetnje smanjile, potrebno je obrezati sliku na širinu RC modela te da dužina ne bude veća od 15cm. Na slici 3.25. prikazana je obrada slike te pravljenje maske.



Slika 3.25. Obrada slike

Vidljivo je da u obzir uzima samo ono što se nalazi u zadanome rasponu te se tako pravi maska kako bi se zanemarile smetnje. Tijekom cijelog rada poželjno je da se crta nalazi u sredini kako bi orjentacija bila najtočnija. Zbog toga što se koristi gotova platforma poželjno je da zavoji budu što blaži kako bi model uspio skretati i pratiti crtu.

Program se sastoji od više dijelova. Prvi dio je dohvaćanje biblioteka, i definiranje raspona boja. Kako u početku nije bilo bežične konekcije, za obradu slike koristio se video. Kao što je prikazano na slici 3.25. vidi se da je korišten raspon tamnih boja. Za izradu staze isprintana je crna crta na bijelom A4 papiru kako bi se ostvario nabolji kontrast. Na slici 3.26. prikazana je početna staza koja je korištena u videu. Bitno je napomenuti da širina crte nije bitna, jer se cijeli princip zasniva na prepoznavanju boja.

```
up_color = 90                                                    # siva boja
low_color = 0                                                    # crna boja
video = cv2.VideoCapture(0)
isManual = True
```

Slika 3.26. Definiranje raspona boja

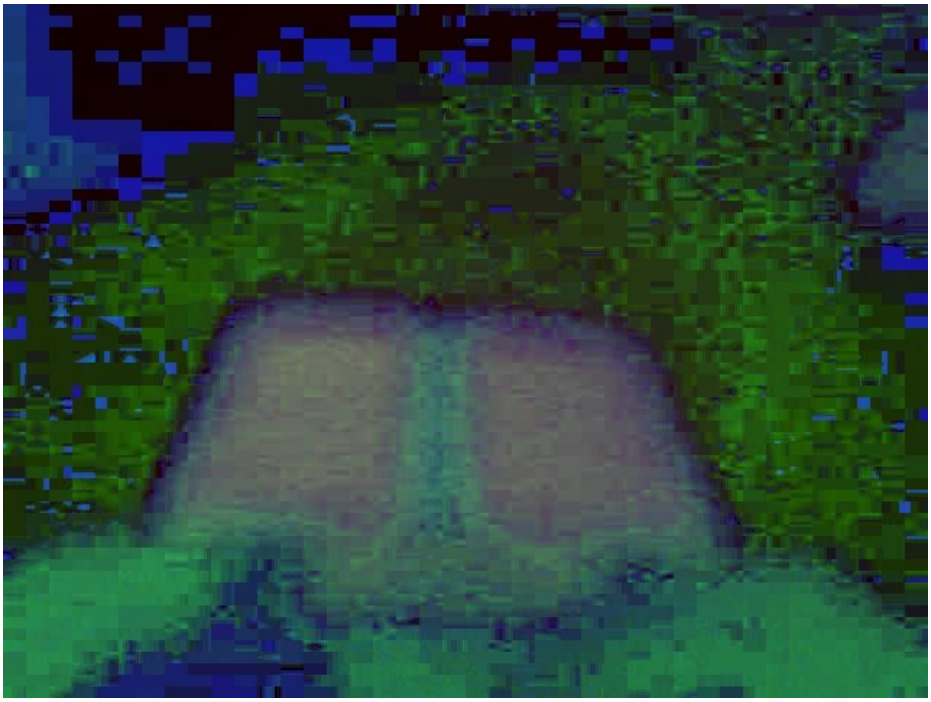
Slika 3.27. *Line tracking* staza

U dugom dijelu programa obrađuje se slika. Uzima se jedan *frame*, te se on zamagljuje kako bi dobili bolji kontrast boja. Kao što je već prije napomenuto, *frame* je potrebno obrezati kako Raspberry ne bi uzimao u obzir predmete iz okoline. Nakon obrade slike pravi se maska.

```
ret, frame = video.read()
if frame is None:
        continue

blurred_image = cv2.GaussianBlur(frame, (7,7), 0)
gray_image = cv2.cvtColor(blurred_image, cv2.COLOR_BGR2GRAY)

mask = cv2.inRange(gray_image, low_color, up_color)
eroded_img = cv2.erode(mask, None, iterations = 3)
_, contours, hierarchy = cv2.findContours(eroded_img.copy(), 1, cv2.CHAIN_APPROX_NONE)
```

Slika 3.28. Obrada slike

Slika 3.29. *Frame* nakon zamagljivanja



Slika 3.30. Izrada maske

U zadnjem dijelu programa ovisno o maski i crti upravlja se motorima. Prvo se prave konture bijele crte te se zatim određuje x i y os. Ukoliko je crta otišla desno od sredine, motori skreću u desno, i obratno, odnosno u lijevo. Na kraju ukoliko nema crte za detekciju program ispisuje „NO LINES“ i prestaje s radom.

```
try:
        contour_x = int(moments['m10']/moments['m00'])
        contour_y = int(moments['m01']/moments['m00'])
except ZeroDivisionError:
        continue

if contour_x >= 210:
        print(contour_x, " | RIGHT\r")
        motor_stop()
        sleep(0.01)
        motor_right()
        motor_forward()

if contour_x < 210 and contour_x > 100:
        print(contour_x, " | FORWARD\r")
        motor_stop()
        sleep(0.01)
        motor_forward()

if contour_x <= 100:
        print(contour_x, " | LEFT\r")
        motor_stop()
        sleep(0.01)
        motor_left()
        motor_forward()
else:
       print("NO LINES\r")
       motor_stop()
```

Slika 3.31. Upravljanje radom motora

Program za udaljeno upravljanje je puno jednostavniji. Program učitava stanje na tipkovnici, odnosno koja je tipka pritisnuta, te ovisno o tome upravlja motorima. Zbog toga što se čita logičko stanje s tipkovnice, motori će ostati u radu, pa je potrebno dodati tipku koja će zaustaviti rad motora što je jedina mana ovakvog sustava udaljenog upravljanja.

```
while True:
        try:
                if not is_obstacle:
                        if isManual is True:
                                char = screen.getch()

                                if char != -1:
                                        print("Rucno (znak): " + str(char) + "\r")

                                if char == 259:                         # ravno
                                        motor_forward()
                                elif char == 258:                       # nazad
                                        motor_reverse()
                                elif char == 260:                       # lijevo
                                        motor_left()
                                elif char == 261:                       # desno
                                        motor_right()
                                elif char == 115:                       # stop (S)
                                        motor_reset()
                                elif char == 109:                       # manual (M)
                                        motor_reset()
                                        isManual = False
                                        continue
                        else:
                                ##motor_stop()
                                char = screen.getch()
```

Slika 3.32. Određivanje smjera upravljanja

3.4. Integracija video veze

Kao što je već spomenuto, video veza je osnovni dio ovog projekta jer se u svakom trenutku može vidjeti okolina sustava. Izvedba video veze je veoma jednostavna, na platformi se nalazi kamera s odašiljačem, a prijamnik se spaja na TV te je time omogućena komunikacija. Problem je taj što je u pitanju analogna veza pa su smetnje učestale, a kanal nije konstantan već se prilikom svakog spajanja mora tražiti kanal. Iako prijamnik koristi napajanje 7-18V, premostavanjem priključka na USB priključak omogućeno je napajanje iz USB priključka na TV-u u iznosu 5V. Na slici 3.28. prikazan je prijamnik RC832 koji se koristi za ovaj projekt prilikom spajanja na kanal 31, a na slici 3.29. prikazana je video veza između sustava i prijamnika.

Slika 3.33. Primanje signala

Slika 3.34. Povratna video veza

4. TESTIRANJE

U ovom poglavlju provodit će se testiranje sustava. Testirat će se postotak uspješnosti prolaska sustava kroz 5 različitih staza.

4.1. Metodologija testiranja

Ideja testiranja je ta da se pomoću A4 papira na kojem je isprintana crta naprave različite staze po kojima će se model kretati. Svakim idućim testiranjem staza će biti zahtjevnija te će biti većih zavoja. Zbog ograničenosti kamere i kretanja modela postoji mogućnost da model izgubi crtu te da stane. Testiranje se sastoji od 5 različitih staza na kojima će se provesti 10 mjerenja te će se zatim računati postotak uspješnosti prolaska.

4.2. Rezultati i interpretacija rezultata

Slika 4.1. Prva staza

1.	Prošao
2.	Prošao
3.	Prošao
4.	Nije prošao
5.	Prošao
6.	Prošao
7.	Nije prošao
8.	Prošao
9.	Prošao
10.	Prošao

Tablica 4.1. Rezultati prvog testiranja

Slika 4.2. Druga staza

1.	Prošao
2.	Nije prošao
3.	Prošao
4.	Prošao
5.	Prošao
6.	Prošao
7.	Nije prošao
8.	Prošao
9.	Prošao
10.	Prošao

Tablica 4.2. Rezultati drugog testiranja

Slika 4.3. Treća staza

1.	Prošao
2.	Prošao
3.	Nije prošao
4.	Prošao
5.	Prošao
6.	Prošao
7.	Prošao
8.	Nije prošao
9.	Nije prošao
10.	Prošao

Tablica 4.3. Rezultati trećeg testiranja

Slika 4.4. Četvrta staza

1.	Nije prošao
2.	Nije prošao
3.	Prošao
4.	Prošao
5.	Prošao
6.	Nije prošao
7.	Prošao
8.	Nije prošao
9.	Nije prošao
10.	Prošao

Tablica 4.4. Rezultati četvrtog testiranja

Slika 4.5. Peta staza

1.	Nije prošao
2.	Prošao
3.	Nije prošao
4.	Nije prošao
5.	Prošao
6.	Nije prošao
7.	Prošao
8.	Nije prošao
9.	Nije prošao
10.	Nije prošao

Tablica 4.5. Rezultati petog testiranja

Nakon provedenog testiranja računa se uspješnost prolaza. Uspješnost prolaza računa se prema formuli 4-1.

$$X = \frac{BROJ\ USPJEŠNIH\ MJERENJA}{BROJ\ MJERENJA} * 100\% \quad (4\text{-}1)$$

$$X_1 = \frac{8}{10} * 100\% = 80\%$$

$$X_2 = \frac{8}{10} * 100\% = 80\%$$

$$X_3 = \frac{7}{10} * 100\% = 70\%$$

$$X_4 = \frac{5}{10} * 100\% = 50\%$$

$$X_5 = \frac{3}{10} * 100\% = 30\%$$

Nakon računanja uspješnosti prolaza može se vidjeti da sustav ne reagira dobro na zavoje. Zbog svoje brzine ne stigne reagirati na vrijeme te ode van staze. Također je problem u okolini modela jer ukoliko ima tamnih objekata u blizini model će krenuti prema njima. Zbog toga što je crta printana laserskim printerom, veliki je stupanj odbijanja svjetlosti pa model nekada ne može prepoznati crtu.

5. ZAKLJUČAK

Samoupravljivi sustavi i sustavi s udaljenim upravljanjem nužni su za današnji svijet. Koriste se svakodnevno kako bi svojim radom olakšali posao čovjeka. Zbog svoje pouzdanosti i sigurnosti u budućnosti će svoju primjenu naći u svim slojevima društvenog i radnog života.

Projekt koji je rađen samo je jedan od primjera samoupravljivih sustava koji pomoću svojih senzora i logičkih sklopova mogu zamjeniti čovjeka. U današnje vrijeme industrija je nezamisliva bez mikroupravljača jer bi čitav proces obrade neke sirovine bio daleko kompliciraniji i nepouzdaniji da njime upravlja čovjek. Samoupravljivi RC model pokazuje da je moguće upravljati autom pomoću informacija iz okoline, ali je i puno pouzdaniji od čovjeka jer brže reagira na opasnosti i prepreke. Za detekciju udaljenih predmeta koristi ultrazvučni senzor koji ima veliku brzinu odašiljanja i primanja signala tako da ima veoma brzo vrijeme odziva. Njegova konstrukcija je veoma jednostavna jer koristi gotove komponente, ali je programski dio veoma zahtjevan jer se moraju eliminirati sve smetnje iz okoline kako bi sustav bio u potpunosti rasterećen i usredotočen na bitne stvari.

Nakon izrade projekta provedena su testiranja RC modela na različitim stazama. Vidljivo je da se povećanjem broja zavoja i njihovog nagiba smanjuje uspješnost prolaza zbog tromosti auta jer funkcionira po principu tenka, prilikom skretanja jedan par motora ide prema naprijed, a drugi u iza pa ne stigne pravilno skrenuti.

Također se kao veliki nedostatak pokazala i mini FPV kamera. Iako su zbog visoke frekvencije kašnjenja minimalna (zanemariva), pojavljuju se konstantne smetnje i promjena kanala što je u trenutcima udaljenog upravljanja nedopustivo.

LITERATURA

[01] Upravljanje modelom vozila putem računalne mreže i Raspberry Pi računala, https://www.veleri.hr/arhiva/files/datoteke/page_privitak/UPRAVLJANJE_MODELOM_VOZILA_PUTEM_RACUNALNE_MREZE_I_RASPBERRY_PI_RACUNALA.pdf (Pristup 07.06.2018.)

[02] PBS, http://www.pbs.org/tesla/ins/lab_remotec.html (Pristup 07.06.2018.)

[03] Bluebird marine systems, http://www.bluebird-electric.net/Bluebird_Boats_Ships_Systems/Nikola_Tesla.htm (Pristup 07.06.2018.)

[04] Sustav za daljinsko djelovanje, http://www.pfst.hr/~ivujovic/stare_stranice/pdf_zip_word/POGL8.pdf (Pristup 10.06.2018.)

[05] How to mehatronics, https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/ (Pristup 10.06.2018.)

[06] First-person view (radio control), https://en.wikipedia.org/wiki/First-person_view_(radio_control) (Pristup 11.06.2018.)

[07] The MagPi, https://www.raspberrypi.org/magpi/pi-zero-w/ (Pristup 11.06.2018.)

[08] Ultrasonic ranging moduce: HC-SR04 http://www.otpornik.com/blog/wp-content/uploads/2013/03/HC-SR04.pdf (Pristup 15.06.2018.)

[09] IBEX, http://www.raspberry-projects.com/pi/pi-hardware/raspberry-pi-zero/raspberry-pi-zero-hardware-general-specifications (Pristup 15.06.2018.)

[10] Secure Shell, https://hr.wikipedia.org/wiki/Secure_Shell (Pristup 18.06.2018.)

[11] Actuator, https://en.wikipedia.org/wiki/Actuator (Pristup 20.06.2018.)

SAŽETAK

U ovom završnom radu rađen je projekt s povratnom video vezom te mogućnosti udaljenog upravljanja pomoću osobnog računala ili mogućnosti pametnog upravljanja pomoću obrade slike. Detaljno je opisan postupak izrade same makete te izrade programa. Glavni problem bio je program za line tracking koji je uz pomoć Python programskog jezika uspješno rješen. Kamera šalje sliku na Raspberry Pi te se zatim ta slika obrađuje, uzima se u obzir samo što se nalazi u zadanome rasponu boja, pravi se maska, a ostalo se zanemaruje. Udaljeno upravljanje omogućeno je jednostavnom „if“ petljom. Prvo je potrebno deklarirati stanja motora, odnosno kada motori idu kojem smjeru i zatim ukoliko je pritisnuta određena tipka motor ide u zadanome smjeru. Povratna video veza ostvaruje se pomoću analogne mini fpv kamere i prijamnika koji je spojen na TV. Problem analognog prijenosa je taj da kanal nikada nije konstantan, a smetnje su učestale.

ABSTRACT

In this thesis a project with live stream and the possibility of remote controlling with a personal computer or the possibility of smart self remote using picture recognision waws made. The procedure of making the model and program was explained in detail. The main problem was the line tracking part which has been successfully resolved with the help of python programming language. The camera sends the image on Rasperry Pi and the image is processed, taken into account only in the given color range, making a mask, and the other is neglected. The remote control is enabled with a simple "if" loop. First it is necessary to declare the status of the motor, and if the right key is pressed, the motor will start to spin. The live stream is enabled with an analog mini fpv camera and reciever which is connected to a TV. The problem with an analog stream is that the channel is never the same and the interference are common.

ŽIVOTOPIS

Leonardo Markotić rođen je 08.12.1996. u Essen, Njemačka. 1999. godine doseljava se u Koprivnicu gdje 2011. godine završava osnovnu školu „Braća Radić“. Potom upisuje Obrtničku školu „Koprivnica“ , smjer Tehničar za računalstvo koju završava s vrlo dobrim uspjehom. Nakon završene srednje škole, 2015. godine u Osijeku upisuje Fakultet elektrotehnike, računalstva i informacijskih tehnologija, smjer Automatika.

PRILOZI

```
#importanje modula
import cv2
import numpy as np
import curses
import RPi.GPIO as gpio
from time import sleep, time
from threading import Thread


# Globalne varijable – vrijednosti poznate u čitavome programu
is_obstacle = False
sensor_started = False


# Pinovi – definiranje GPIO pinova
LIJEVO_NAPRIJED = 15
LIJEVO_IZA = 11


DESNO_NAPRIJED = 16
DESNO_IZA = 18


TRIG = 37
```

```
ECHO = 38


# GPIO inicijalizacija

gpio.setmode(gpio.BOARD) #postavljanje BOARD načina određivanja pinova(piše se broj pina, a ne broj GPIO pina)

gpio.setwarnings(False) #onemogućavanje greški


gpio.setup(TRIG, gpio.OUT) #inicijalizacija pinova

gpio.setup(ECHO, gpio.IN)

gpio.output(TRIG, False)


gpio.setup(LIJEVO_NAPRIJED, gpio.OUT)

gpio.setup(LIJEVO_IZA, gpio.OUT)


gpio.setup(DESNO_NAPRIJED, gpio.OUT)

gpio.setup(DESNO_IZA, gpio.OUT)


gpio.output(LIJEVO_NAPRIJED, False)

gpio.output(LIJEVO_IZA, False)

gpio.output(DESNO_NAPRIJED, False)

gpio.output(DESNO_IZA, False)
```

```
forward_left_pwm.gpio.PWM(LIJEVO_NAPRIJED, 50) #dodavanje novih varijabli s PWM-om i zadanim vrijednostima

forward_right_pwm.gpio.PWM(DESNO_NAPRIJED, 50)


reverse_left_pwm.gpio.PWM(DESNO_IZA, 50)

reverse_right_pwm.gpio.PWM(LIJEVO_IZA, 50)


forward_left_pwm.start(0) #pokretanje PWM-a

forward_right_pwm.start(0)


reverse_left_pwm.start(0)

reverse_right_pwm.start(0)


sleep(0.5)


#Senzor


def check_distance(): #provjera udaljenosti predmeta od senzora

    global is_obstacle, sensor_started

    distance = 0
```

```
        while sensor_started:
                gpio.output(TRIG, True)#deklaracija stanja senzora
                sleep(0.00001)
                gpio.output(TRIG, False)

                while gpio.input(ECHO) == 0;#nema prepreke, mjeri vrijeme
                        pulse_start = time()

                while gpio.input(ECHO) == 1;#očitavanje prepreke, zasustavlja mjerenje vremena
                        pulse_end == time()

                pulse_duration = pulse_end - pulse_start#računa vrijeme rada senzora
                distance = round(pulse_duration * 17150, 2)#računa udaljenost
                """
                if distance <= 60;#uspoređuje udaljenost
                        print("Distance: " + str(distance) + "\r")#ukoliko je očitao prepreku ispisuje udaljenost od prepreke
                        is_obstacle = True
                else:
                        is_obstacle = False""" #ukoliko nema prepreke ništa se ne događa
```

```
# Inicijalizacija tipkovnice
screen = curses.initscr()
curses.noecho()
curses.cbreak()
screen.nodelay(True)
screen.keypad(True)

# Funkcije
def motor_forward_manual():#određivanje kombinacije pinova za upravljanje prema naprijed s PWM-om prilikom udaljenog upravljanja
        forward_left_pwm.ChangeDutyCycle(90)
        forward_right_pwm.ChangeDutyCycle(90)
        reverse_left_pwm.ChangeDutyCycle(0)
        reverse_right_pwm.ChangeDutyCycle(0)

def motor_forward_line():#određivanje kombinacije pinova za upravljanje prema naprijed s PWM-om prilikom praćenja crte
        forward_left_pwm.ChangeDutyCycle(40)
        forward_right_pwm.ChangeDutyCycle(40)
        reverse_left_pwm.ChangeDutyCycle(0)
        reverse_right_pwm.ChangeDutyCycle(0)
```

```
def motor_reverse():#određivanje kombinacije pinova za upravljanje prema natrag s PWM-om
        forward_left_pwm.ChangeDutyCycle(0)
        forward_right_pwm.ChangeDutyCycle(0)
        reverse_left_pwm.ChangeDutyCycle(90)
        reverse_right_pwm.ChangeDutyCycle(90)

def motor_right():#određivanje kombinacije pinova za upravljanje u desnu stranu s PWM-om
        forward_left_pwm.ChangeDutyCycle(50)
        forward_right_pwm.ChangeDutyCycle(0)
        reverse_left_pwm.ChangeDutyCycle(0)
        reverse_right_pwm.ChangeDutyCycle(50)

def motor_left():#određivanje kombinacije pinova za upravljanje u lijevu stranu s PWM-om
        forward_left_pwm.ChangeDutyCycle(0)
        forward_right_pwm.ChangeDutyCycle(50)
        reverse_left_pwm.ChangeDutyCycle(50)
        reverse_right_pwm.ChangeDutyCycle(0)

def motor_stop():#određivanje kombinacije pinova za zaustavljanje motora
        forward_left_pwm.ChangeDutyCycle(0)
```

```
        forward_right_pwm.ChangeDutyCycle(0)
        reverse_left_pwm.ChangeDutyCycle(0)
        reverse_right_pwm.ChangeDutyCycle(0)

        gpio.output(LIJEVO_NAPRIJED, False)
        gpio.output(LIJEVO_IZA, False)
        gpio.output(DESNO_NAPRIJED, False)
        gpio.output(DESNO_IZA, False)

# Inicijalizacija kamere
up_color = 90                                   # siva boja
low_color = 0                                   # crna boja
video = cv2.VideoCapture(0)
isManual = True

#Pozivanje senzora
sensor_started = True
sensor_thread = Thread(target = check_distance args = ())
sensor_thread.start()
```

```
# Petlja
while True:
    try:
        if not is_obstacle: #ukoliko nema prepreke

        if isManual is True: #ukoliko je na ručnom upravljanju
            char = screen.getch()

            if char != -1:#ukoliko nije unesen niti jedan char u konzolu
                    #prikazuje poruku samo kada je nešto uneseno
                print("Rucno (znak): " + str(char) + "\r")

            if char == 259:                 #ukoliko je pritisnuta gornja strelica
                motor_forward_manual()      # ravno
            elif char == 258:               #ukoliko je pritisnuta donja strelica
                motor_reverse()             # nazad
            elif char == 260:               #ukoliko je pritisnuta lijeva strelica
                motor_left()                # lijevo
            elif char == 261:               #ukoliko je pritisnuta desna strelica
                motor_right()               # desno
            elif char == 115:               # stop (S)
```

```
                motor_stop()
            elif char == 109:                    # manual (M)
                isManual = False        #ukoliko je pritisnuta tipka “M“ prebaciva
se na praćenje crte
                continue
        else:
            ##motor_stop()
            char = screen.getch()
            if char != -1:
                print("Line tracking (znak): " + str(char) + "\r")

            if char == 109:          # manual (M) #prebacivanje na ručno upravljanje
                motor_reset()
                isManual = True
                continue

# Prepoznavanje linije
ret, frame = video.read() #dohvaćanje slike s kamere
if frame is None:
    continue
```

```
            blurred_image = cv2.GaussianBlur(frame[150:510, 150:500], (7,7), 0)
        #zamagljivanje slike i obrezivanje

            gray_image = cv2.cvtColor(blurred_image, cv2.COLOR_BGR2GRAY)#obrada
slike

            mask = cv2.inRange(gray_image, low_color, up_color)#ukoliko se nalazi u
rasponu boja prvi se maska

            eroded_img = cv2.erode(mask, None, iterations = 3)

            _, contours, hierarchy = cv2.findContours(eroded_img.copy(), 1,
cv2.CHAIN_APPROX_NONE) #pravljenje kontura


            if(len(contours) > 0):#postavljanje maksimalnih vrijednosti kontura

                max_contours = max(contours, key=cv2.contourArea)

                moments = cv2.moments(max_contours)


            try:

                contour_x = int(moments['m10']/moments['m00'])#određivanje x i y osi

                contour_y = int(moments['m01']/moments['m00'])

                except ZeroDivisionError:

                continue


                if contour_x >= 210:#određianje položaja konture za upravljanje u desnu
stranu

                        print(contour_x, " | RIGHT\r")
```

```
                    motor_stop()
                    motor_right()

                if contour_x < 210 and contour_x > 175: #određianje položaja konture za upravljanje ravno
                    print(contour_x, " | FORWARD\r")
                    motor_forward_line()

                if contour_x <= 175: #određianje položaja konture za upravljanje u lijevu stranu
                    print(contour_x, " | LEFT\r")
                    motor_stop()
                    motor_left()
                else:
                    print("NO LINES\r") #ukoliko nema ništa što se nalazi u rasponu, motor se zaustavlja i ispisuje se „NO LINES“
                    motor_stop()

        else: #ukoliko nema interrupta, motor se gasi
            motor_stop()
```

```
        except KeyboardInterrupt:
                break

# Čišćenje varijabli
sensor_started = False
sensor_thread.join()

curses.nocbreak();
screen.keypad(0);
screen.nodelay(False)
curses.echo()
curses.endwin()

motor_stop()
video.release()
gpio.cleanup()

print("Program exit\r")
```